Seattle University School of Law Digital Commons

Faculty Scholarship

1-1-2012

Many-to-Many Contracts

Heidi S. Bond

Follow this and additional works at: https://digitalcommons.law.seattleu.edu/faculty

 Part of the Contracts Commons

Recommended Citation

Heidi S. Bond, *Many-to-Many Contracts*, 86 TUL. L. REV. 519 (2012).

This Article is brought to you for free and open access by Seattle University School of Law Digital Commons. It has been accepted for inclusion in Faculty Scholarship by an authorized administrator of Seattle University School of Law Digital Commons. For more information, please contact coteconor@seattleu.edu.

TULANE LAW REVIEW

VOL. 86 FEBRUARY 2012 NO. 3

Many-to-Many Contracts

Heidi S. Bond*

In classical contract law, the concept of one-to-one negotiations is familiar: contracts where one party negotiates with the other and, eventually, terms are offered and then accepted. More modern times have made us comfortable with the notion of one-to-many contracts: contracts typically drafted by large corporations and then distributed on a take-it-or-leave-it basis to the masses. This Article discusses a third kind of contract: a many-to-many contract, which may look like the standard one-to-many contract in that it is composed of nonnegotiable language. But when the arrangements between the parties are further considered, we will see that the point of the contract is not for one party to impose its terms on another without question, but for a series of parties to determine the best way to facilitate interchange. In such situations, imposing standard doctrines of contract interpretation may frustrate the purpose of the contract entirely, and for that reason, courts should approach the interpretation of these contracts with care.

* © 2012 Heidi S. Bond. Assistant Professor, Seattle University School of Law. I would like to thank Rebecca Eisenberg, Don Herzog, Omri Ben-Shahar, Maggie Chen, Adam Blumenkrantz, Alaina Fotiu-Wojtowicz, Nadine Gartner, Matt Morris, Rebecca McNeill, Matt Rojas, Kim Vu, Moran Yahav, Laura Yockey, and Chris Zealand for helpful comments and criticism.

HeinOnline -- 86 Tul. L. Rev. 519 2011-2012

I. INTRODUCTION

When interpreting a contract, courts start from the personal and the specific (for instance, communications between the parties that elucidate the meaning of certain terms or evidence of a course of prior dealings[1]) and work up to the impersonal and the general (such as dictionary definitions and trade usage[2]). This is hardly surprising. After all, the goal in contract interpretation is to determine whether there has been a meeting of the minds, and if so, where the minds have met.[3] Evidence that is specific to a negotiation sheds light on what the parties actually intended in a way that evidence of general interpretation does not.

Of course, it is also understood that in the modern world, offer and acceptance rarely come attached to the sort of meeting of the minds that is idealized in law school contractual cases. Actual handshakes and negotiations are rare. Goods and services are often accompanied by nonnegotiable boilerplate. Parties regularly sign off on these forms without reading anything beyond the dollar amount on the first page of the contract—and they certainly never delve into the dense morass of conditions that follows. There is no reason for parties to come to even a modest understanding of those terms, after all; if you want cell phone service, electricity, or life insurance, you pay up and pray that the conditions you ignored do not end up working against you.

Under these circumstances, most people presented with boilerplate contracts rarely form anything that bears even a faint resemblance to an *intent* as to the meaning of the text. Given these circumstances, courts impute intent where no conscious impetus ever existed. This is not to say that courts wave a white flag at the age-old doctrine of intent in contract law, surrendering all hope of compliance. Instead, the court invokes something of a meta-intent: the parties

1. RESTATEMENT (SECOND) OF CONTRACTS § 4 cmt. a (1981).
2. *Id.* § 201 cmt. a.
3. *Id.* § 201 cmt. e.

HeinOnline -- 86 Tul. L. Rev. 520 2011-2012

believed that hashing out the details of the contract wording would not be worth the time and attention and so have effectively delegated to the courts the task of determining the precise meaning of contractual language. In other words, the intent of the parties is not to form an intent as to every line of the contract, but to trust that the general rules—dictionary definitions, industry meanings, and, if all else fails, *contra proferentem*—will be sufficiently to their benefit so that the contract can be ignored.

But even though general rules typically govern form contracts, the specific will still trump the general where it is available. If an insured asks his insurer the meaning of a particular provision, the response given will shed substantial light in any subsequent action.

It may seem that these two scenarios cover the entire space of all possible contracts. There is the idealized agreement hashed out between two parties, where there is a true meeting of the minds. This is the type of contract that I will call a "one-to-one" contract, because one party negotiates with the other. And then there is the equally familiar "one-to-many" contract: contracts typically drafted by one larger entity—an insurer or a service provider—and distributed on a take-it-or-leave-it basis to the masses.

But there is a third kind of contract.

This Article tackles a puzzle of fairly recent vintage. To illustrate the problem, imagine a bond investor. She is considering purchasing bonds issued by a particular company. Assume for the sake of this example that both she and the companies she considers are fairly sophisticated parties, and that the bonds, when originally issued, were intended to be bought and sold on the bond market. In deciding whether she wants to purchase a particular bond issue, she will want to know about the financial health of the corporation. She will want to know how previous bonds have performed. And, of course, she will need to know the terms of the issue that she is considering purchasing.

But how can she evaluate those terms? The only evidence she will have available to her is the paper that accompanies the bond issue. In other words, she will only have evidence of *general* intent—plain contract language, as seen through the filter of dictionary definitions, trade usage, and the like. She will, however, lack any *specific* evidence of contract meaning. Unless she is closely connected with the person who originally purchased the bonds from the issuing company, she will never know if the original purchaser of the bonds asked for clarification of specific terms of the bond. She will not know if the parties had a prior course of dealings that might shed light

HeinOnline -- 86 Tul. L. Rev. 521 2011-2012

on the meaning of terms. In other words, she will have access to only the general terms of the contract—the black and white text—stripped of all specifics.

Suppose, in addition, that she wants to buy bonds—identically worded, with identical terms, from the same exact company—from two sources. One of those sources asked the company to clarify the meaning of a term. The other did not. Is it conceivable that those bonds could potentially have different terms, despite their facial identity?

What if our hypothetical bond investor then sells bonds from both sources as a group to a second investor? If the specific trumps the general in this circumstance, bond investors will need to do more than examine the language of a contract. They would have to know the history of the contract negotiations from the beginning. They would have to trace the path of a bond, know who purchased it, and know what negotiations were involved in its purchase. And they would not be able to assume that facially identical bonds were, in fact, identical. In short, the easy exchange and valuation of such bonds would be quickly frustrated—a result that neither party intended.

This short example is one of a subset of what I have called "many-to-many" contracts. Many-to-many contracts are contracts where the initial parties to the contract need not be the parties who bring the language into court—and in fact, they rarely will be. They are contracts where evidence of specific interactions that arise during the initial contract formation will rarely seem relevant to interpretation of the contract. Most specifically, they are contracts where both parties enter into the arrangement with the intent to create a contract that can be passed from many people to many others.

Part II of this Article discusses why specific evidence of contract formation always trumps general evidence in the case of one-to-one and one-to-many contracts. It also discusses a specific rule that applies to boilerplate contracts—namely, the doctrine of *contra proferentem*—and explains how these rules lead to nonuniform interpretation of contracts in the one-to-many context.

The rule that ambiguities should be construed against the drafter is, of course, particularly strong when applied to boilerplate contracts. The rationale is twofold. First, the party that did not provide the language probably did not contemplate events relating to terms that

HeinOnline -- 86 Tul. L. Rev. 522 2011-2012

could not be changed.[4] As such, it cannot be charged with the responsibility—or the burden—of avoiding ambiguity in drafting.[5] Second, courts attempt to motivate the drafting party to avoid ambiguity by placing the burden of unclear language on its shoulders.[6]

Part III argues that not all form contracts are created equal. I revisit the rules of contract construction discussed in Part II and demonstrate using a simple example of a many-to-many project—namely, a very early stage of a project known as Rockbox. The Rockbox project is, of course, one of literally thousands of software projects that are released under the many-to-many contract known as the General Public License (GPL). I chose it because it is relatively self-contained, and in its early years, it involved only a very small number of people. In other words, it is a many-to-many project where the initial "many" is the relatively tiny number of thirty-eight. Part III demonstrates that, in the many-to-many context, the common rules of contract construction that were explored in Part II, which lead to nonuniformity, would actually frustrate the intent of the parties.

Part IV offers a solution to the puzzle presented in Part III. Rules that lead to nonuniformity should not be applied in the many-to-many context. Unlike the one-to-many context, where it is possible that many understandings of the same text could be considered in a court, as long as the parties differed, the hallmark of a many-to-many contract is that it carries the strong inference (and in some cases, the explicit statement) that the parties intended a uniform interpretation among all contracting parties in order to facilitate exchange.

Part IV contends that this modified many-to-many rule of construction is already in play in the interpretation of bond issue agreements and should be extended to other many-to-many circumstances.

One final comment about the scope of many-to-many contracts: I have chosen relatively simple examples to explain this problem. But many-to-many contracts are not limited to small software projects centered in Sweden or bond issues available only to sophisticated parties. The GPL is an example of an attempt to contract around default intellectual property rules. But it is not alone. As distance

4. *Id.* § 206; 5 MARGARET N. KNIFFIN, CORBIN ON CONTRACTS § 24.27 (Joseph M. Perillo ed., rev. ed. 1998).

5. E. ALLAN FARNSWORTH, CONTRACTS § 7.11 (3d ed. 1999).

6. *E.g.*, United States v. Turner Constr. Co., 819 F.2d 283, 286 (Fed. Cir. 1987) ("*Contra proferentem* . . . correctly requires the drafter to use care and completeness in creation of a contract.").

HeinOnline -- 86 Tul. L. Rev. 523 2011-2012

shrinks and communities form across greater distances, contracts are increasingly used to form communities that rely upon altered default rules.

Rules of construction that apply with particular force to one-to-many contracts are the norm. The contract drafted from scratch is, in modern times, a rarity. But one-to-many contracts are distinct from many-to-many contracts. This Article suggests that courts should distinguish between preprinted contracts that save private drafting time by automating the negotiation process and form contracts that standardize terms in order to achieve easily transferrable uniformity. And while public policy and basic common sense support this result, the reason to do this, after all, is the oldest one in the book: It is what the parties wanted.

II. ONE-TO-ONE AND ONE-TO-MANY CONTRACTS: HOW RELATIONSHIPS AMONG PARTIES DETERMINE CONTRACT CONSTRUCTION

The ultimate goal in contract interpretation is to give effect to the intent of the parties. This Part discusses how, in both one-to-one and one-to-many contracts, identical contract language can give rise to differing interpretations.

This is, of course, the nature of the contractual beast. Because contracts are private agreements, nobody expects the contracts to bear any relation to each other. Thus, for instance, two people may discuss the terms of a contract for furniture and agree that payment is due on "delivery." They may even discuss what constitutes "delivery"—whether it is the furniture delivered and assembled inside the house, or merely left on the driveway in a heap of compressed wood, screws, and impenetrable instructions. It is perfectly plausible that a furniture maker could have two contracts with two different people, both allowing for payment on "delivery"—but because the parties have reached different understandings as to the meaning of "delivery," the term would be interpreted two different ways. After all, it is the specific understanding between the parties that must control the meaning of the word. The meaning of the contract is thus one-to-one: it depends on the specific interactions that one party had with the other.

This does not change much in the modern contract. In most modern contract disputes, the intent of the parties may not be explicitly discussed. It is probably not even explicitly considered. When modern form contracts are signed, the parties rarely discuss the

meaning of potentially ambiguous terms.[7] They very often do not ask what it means to subrogate rights. In part, this is because the person presenting the contract rarely has the authority to change the contractual language. But it is also because modern form contracts are larded with clauses, subclauses, sections, and legalistic language. The rare person who looks over the terms of a contract often has no idea what they mean.[8] How, then, do courts determine what the parties really intended to achieve with the contract language?

The simple answer is that they do not. When a boilerplate contract is supplied by one party, courts construe ambiguities against the party that provided the language.[9] To some small extent, this rule tracks party intent. The court assumes that the drafting party should have contemplated all potential meanings of their words.[10] Likewise, courts assume that the party that lacks the opportunity to bicker over wording is less likely to consider how the language could be understood.[11] Finally, they recognize that the parties do not want to be burdened with hashing out such specific details and so rely on courts to apply interpretive doctrines. So in the final analysis, the courts will impute to both the drafting and the nondrafting parties an interpretation that is favorable to the nondrafting party.[12]

Additional rationales include a desire to motivate the drafting party to avoid ambiguity[13] and a desire to stick the party more likely to have greater resources at its disposal with the costs of ambiguity.[14]

For the most part, courts construing contracts are motivated by these policy reasons rather than the traditional contract doctrine of intent. But intent does not trump policy. Regardless of how much the courts may wish that the drafting party had expressed the contract meaning clearly, and regardless of the extent of harm to the plaintiff or the resources available to the defendant, courts insist that only

---

7. Michael I. Meyerson, *The Efficient Consumer Form Contract: Law and Economics Meets the Real World*, 24 GA. L. REV. 583, 594-95 (1990).

8. *Id.* at 595; 5 KNIFFIN, *supra* note 4, § 24.27, at 295 ("The applicant may not even read the policy, being discouraged by the number of terms and the fineness of print.").

9. FARNSWORTH, *supra* note 5, § 7.11, at 473-74; 5 KNIFFIN, *supra* note 4, § 24.27, at 282-83.

10. 5 KNIFFIN, *supra* note 4, § 24.27, at 283.

11. *Id.*

12. *Id.*

13. *Id.*; United States v. Turner Constr. Co., 819 F.2d 283, 286 (Fed. Cir. 1987).

14. *See* Daniel D. Barnhizer, *Inequality of Bargaining Power*, 76 U. COLO. L. REV. 139, 147-48 (2005).

HeinOnline -- 86 Tul. L. Rev. 525 2011-2012

reasonable interpretations of contracts can be considered.[15] Everyone knows that nobody reads form contracts. It may seem like a raw deal to deny insurance coverage because of a technicality spelled out in small print. But courts continually insist that the clear and unambiguous language of the contract must control.[16]

Of course, the rule that contracts are construed against the drafter applies whether the drafter wrote the contract for a single deal or employs a one-size-fits-all form contract in all of its standard dealings. But the principal applies with greater force to this slightly more complex one-to-many contract.[17] As to intent, the party that supplies language to thousands should actually be more aware of potential misunderstandings that might arise under the contract; if they are not, the court has no problem adopting a rule that punishes them for their lack of forethought. And, in general, a party that supplies identical language to a whole slew of people chooses boilerplate terms that the other party can protest only by not signing, so a somewhat lackadaisical response on the part of the nondrafting party is perfectly reasonable. Furthermore, when the contract's language is supplied to many people, the supplier of the language should have an even greater stake in tightening contract language. On policy grounds, placing the burden on the only party capable of resolving the ambiguity makes perfect sense. Because those parties are typically more sophisticated, they undoubtedly intend to opt for the additional burdens imposed upon them in the event that the deal should go south, rather than go through the hassle of negotiating each contract individually.

But just as in the one-to-one situation, a one-to-many contract may result in identical terms in identical contracts being construed differently.[18] Imagine the following.

John contracts with Best Insurance. He does not understand the terms. In fact, he does not look at them at all, just initialing on the appropriate X's and donating his incomprehensible signature at the end. John's contract with Best Insurance specifies—unbeknownst to John—that he must report all accidents, even accidents that are below his deductible. After all, Best Insurance wants to know if John is a risky driver, and if he is the sort of person who regularly taps cars

15. *See* FDIC v. Conn. Nat'l Bank, 916 F.2d 997, 1006 (5th Cir. 1990); 5 KNIFFIN, *supra* note 4, § 24.27, at 283-87.

16. 5 KNIFFIN, *supra* note 4, § 24.27, at 283-87.

17. *Id.* § 24.27, at 293-95.

18. For a number of reasons, this is unlikely. *See infra* text accompanying note 22.

HeinOnline -- 86 Tul. L. Rev. 526 2011-2012

because he tailgates, they will want to increase his premiums to reflect the risk.

John, of course, is a lousy driver. One day, after work, he runs into a car that slows down at a yellow light. But it is John's lucky day—the fender is dented and the paint is scratched, but the damage to both vehicles can be fixed by a decent body shop for $600. He pays the bill, never tells his insurance company, and keeps on driving. And, because John is a lousy driver, he does the same thing again. And again. And again. And yet again.

A year later, John (did I mention he is an extremely lousy driver?) is talking on his cell phone and tailgating. Once again, the car in front of him slows for a red light. This time, John sails into the vehicle. He totals his car and the car in front of him, but this time, he approaches his insurance company.[19] Naturally, they object because John failed to inform them of his five prior accidents.

"Those weren't accidents," John splutters. "Those were mere fender benders! They were just taps!"

Unfortunately for John, there is a pretty clear way to define what counts as an accident: Anything that needs to be reported to the police counts as an accident. In his town, that means anything that causes damage over $500 is an accident. John is just plain out of luck, and the courts find against him.

Now consider Jane. She is a particularly savvy individual. Before signing an insurance contract with Best Insurance that is identical to the one John signed (except, perhaps, for the price), Jane talks to her insurance agent.

"I see," Jane says craftily, "that I'm required to inform you about all accidents. What constitutes an *accident*? What if I just tap a neighbor's car while parking and I scratch it? Do I have to report it?"

"No way," says the agent, eager for his commission. "It's not an accident unless there are damages above your $1000 deductible. Otherwise, it's just a tap—a fender bender, so to speak."

Jane may be a savvy individual, but like John, she is also a lousy driver. She, too, bumps fenders with abandon, paying off the other drivers and failing to report the tiny accidents to her insurance company. And just like John, one day she gets in a real accident, totaling three vehicles and damaging four others. She brings a claim against Best Insurance, asking for reimbursement.

---

19. The insurer may not be able to cut John off completely at this point, but it may be able to assess penalties for his failure to report.

"Aha!" says Best. "You were in five accidents before, and you reported none of them!"

"Those?" says Jane. "Those weren't *accidents*. They were fender benders, mere bumper taps. As long as they were less than my deductible, I don't have to report them."

"Nonsense," says Best Insurance. "An accident is defined as anything you have to report to the police, and in this town, you have to report anything above $500."

If Jane had never talked to her insurance agent, the general definition that was used in John's case would apply. But Jane has more specific evidence. She, after all, talked to her agent, and they reached a more specific understanding.

Best, however, refuses to listen. "This term has been examined by courts in the recent case of Best Insurance versus John," they say. "I'm sorry, but the meaning of this term has already been decided, and it's not what you say."

In her most reasonable tone of voice, Jane demands of Best, "What the devil does your contract with *John* have to do with *me*?"

And she has the right of it. Despite sharing the same language—reconciled with a slight shake of our heads as we recognize that identical language may have different meanings for different parties—John's contract has nothing to do with Jane's. The specific trumps the general, and because Jane got specific, and John did not, Jane will prevail, and John will not. Aside from perhaps a frown of cognitive dissonance from Best Insurance, there is little reason to strive for uniformity of interpretation between the two insurance policies.

A similar situation may occur even if neither party gets specific. After all, the rule of *contra proferentem* says that a term, if ambiguous, is to be construed against the insurer.[20] One can imagine situations in which a term with two reasonable definitions could benefit the insured with one definition in one situation, and benefit the insurer in the other.[21]

---

20. *See* 5 KNIFFIN, *supra* note 4, § 24.27, at 282-83.

21. The simplest (and unlikeliest) example of this is as follows. Assuming that all terms in the contract are interpreted identically—not an unlikely assumption, imagine a contract that reimburses for all "accidents" and requires reporting of all "accidents." If the word "accident" is not defined in the policy, one can imagine an insured failing to report a fender bender, and yet simultaneously obtaining compensation for a later car crash, on the grounds that the earlier fender bender did not qualify as an "accident." But it is also possible that someone else can obtain compensation for a mere fender bender—if he chooses to report it to his insurance company. It does not matter that in one circumstance "accident" meant "not a fender bender" and in the other it included fender benders. In such circumstances, the

HeinOnline -- 86 Tul. L. Rev. 528 2011-2012

Of course, these situations rarely come up. Ambiguous contracts can cut against the insurer under either of two reasonable circumstances. Most of the time, the choice is between denying coverage (good for the insurer) or allowing it (bad). For the most part, then, insurance contracts, even if ambiguous, take on a sort of uniformity simply because the question of whether the term is ambiguous is a matter of law. Once a term has been declared ambiguous, its interpretation in favor of the insured usually has the effect of uniformity.

Nonuniformity may occasionally result when identically worded contracts are not both considered ambiguous.[22] This may happen because courts differ as to what constitutes ambiguity, or it may occur because extrinsic evidence may clarify ambiguity in one circumstance and not in another. But in one-to-many contracts, because the drafter usually sits on one side of the table—as for instance insurer, seller, or provider of warranty—it is unlikely that ambiguities will cut both ways.

One final point should be made here. This rule of contract construction does not displace the intent doctrine associated with simple contracts. One-to-many contracts are simply not *intended* to be interpreted uniformly. Companies choose to adopt one-to-many contracts to save drafting time and, perhaps, to calculate risks associated with the contract more accurately. Individuals choose to sign them because nobody is interested in negotiating for small potatoes. Parties may have interests in uniform interpretation—the insurer, and possibly even the insured, in clarifying the scope of

---

insurance company will usually chalk up such variability to the costs of having a form contract.

It is unlikely that this will happen, but a more convoluted example can undoubtedly be constructed.

22. One potential example of this can be found in Ford's 1970s-era contract with its dealers. In one circumstance, paragraph 17(c) of that agreement was held ambiguous in scope. Semmes Motors, Inc. v. Ford Motor Co., 429 F.2d 1197, 1206-07 (2d Cir. 1970). Ford argued that the paragraph in question, which allowed termination only if Ford offered the dealer an opportunity to cure deficiencies, applied only to circumstances listed in a previous section. *Id.* The court found the application of 17(c) ambiguous and ruled that it could include any number of deficiencies, including out-and-out fraud. *Id.* at 1207. By contrast, another court failed to apply the rule of ambiguity with respect to 17(c). Lyon Ford, Inc. v. Ford Motor Co., 342 F. Supp. 1339, 1344 (E.D.N.Y. 1971). Holding that paragraph 17(c) did not apply in insolvency, the court found that Ford did not need to offer a bankrupt dealer an opportunity to cure. *Id.* As far as I can tell, both courts were construing identical contract language. The *Lyon Ford* court does not explain its rationale; it could be that the dealer in *Lyon Ford* failed to argue that the term was ambiguous. Nonetheless, there is tension between *Semmes Motors*' declaration that 17(c) could apply to any reason for termination and *Lyon Ford*'s decision that insolvency did not count.

coverage—but there is rarely anything in the contract language itself that suggests that parties should imagine that their coverage hinges on their neighbor's coverage.

The remaining Parts of this Article examine a number of circumstances where boilerplate language demonstrates that the parties intended that the language between all parties should be treated in a uniform fashion. The slight cognitive dissonance that is so easily dismissed in the one-to-many context becomes a major headache in many-to-many contracts.

III. THE MANY-TO-MANY CONTEXT: HOW STRUCTURAL DIFFERENCES MATTER

One primary hallmark of one-to-many contracts is that we can easily decouple parties into two-party contract pairs. Best Insurance had a contract with John; Best Insurance had a contract with Jane. Best Insurance can have contracts with an infinite number of people, and each of those contracts can take on a myriad of reasonable meanings. While one party may make one thousand identical insurance contracts, there is no inherent connection between the parties except the common language. And despite that common language, we can construe one contract without reference to any other. Specific interactions, or perhaps different applications of the doctrine of *contra proferentem*, all suggest that for the purpose of contract construction, one-to-many contracts can be treated analytically as if they were a series of one-to-one contracts.

But some contracts cannot be easily broken down into two-party pairs. This Part argues that many-to-many contracts are structurally different. Specifically, this Part examines many-to-many contracts, taking as a test case the infancy of the software project known as "Rockbox." It shows how application of the specific before the general would frustrate the purposes of the contract upon which the exchange of intellectual property was predicated.

A few words about the Rockbox project. In 2001, Björn Stenberg became fed up with his Archos MP3 music player. The hardware was wonderful, but the software that ran the player was abysmally buggy, freezing in spots, and incapable of playing songs randomly. But Stenberg was unwilling to wait for Archos to fix its errors in a subsequent version of the project. Instead, he and several of his friends reverse engineered the Archos hardware and wrote an entirely new piece of software for the music player. They called their

HeinOnline -- 86 Tul. L. Rev. 530 2011-2012

new version of the software "Rockbox."[23] Rather than selling it, they made it available to everyone for free.[24] And because they understood that other people lusted after features that had not been included, they allowed anyone to download and modify their project. This "opening" of the Rockbox code allowed other members of the public to add to the Rockbox project. As of this writing, Rockbox is available in forty-seven different languages[25] and for MP3 players produced by at least twelve different manufacturers, including several generations of Apple's iPod.[26] It is capable of reading menus and filenames out loud to visually impaired users,[27] playing games originally intended for Nintendo's Gameboy,[28] and—with the LCD screen set to maximum brightness, broadcasting pure white—functioning as an extremely expensive and inefficient lamp.[29] And, the Rockbox project had the potential for an infinite number of such features to be available as "plugins" long before Apple made its App Store a reality.[30]

At its heart, Rockbox is dependent upon an agreement between the software developers and the users.[31] The private agreement, in this

---

23. *Rockbox*, WIKIPEDIA, http://en.wikipedia.org/wiki/Rockbox#History (last updated Jan. 20, 2012).

24. Entrepreneurs may wonder how Stenberg made money off of Rockbox. The answer in that case is simple: he did not. As baffling as it may seem, he did it for fun and to provide a useful service to the community.

25. *Rockbox Translations*, ROCKBOX, http://translate.rockbox.org/ (last updated Jan. 20, 2012).

26. *See* ROCKBOX, http://www.rockbox.org/ (last updated Jan. 12, 2012). Note that this figure includes unstable ports; if we exclude those, the number of manufacturers is eight, not twelve.

27. *BlindUsersIndex*, ROCKBOX, http://www.rockbox.org/twiki/bin/view/Main/BlindUsersIndex (last updated June 4, 2008).

28. *PluginRockboy*, ROCKBOX, http://www.rockbox.org/wiki/PluginRockboy (last updated May 17, 2010).

29. *PluginLamp*, ROCKBOX, http://www.rockbox.org/wiki/PluginLamp (last updated Sept. 14, 2010).

30. *Compare BlindFAQ*, ROCKBOX, http://www.rockbox.org/wiki/BlindFAQ?rev=1 (last updated June 23, 2004), *and PluginRockboy*, ROCKBOX, http://www.rockbox.org/wiki/PluginRockboy?rev=1 (last visited Jan. 20, 2012) (date of first version of Rockboy plugin is March 2005), *with* Mathew Honan, *Apple Unveils iPhone*, MACWORLD (Jan. 9, 2007, 1:55 PM), http://www.macworld.com/article/54769/2007/01/iphone.html (iPhone first released in 2007).

31. This statement is, in many ways, too pat. There is some controversy over whether the General Public License (GPL) is a bare license or a contract. *See* Jason B. Wacha, *Taking the Case: Is the GPL Enforceable?*, 21 SANTA CLARA COMPUTER & HIGH TECH. L.J. 451 (2005). To sum up the problem: Most software licenses are both licenses *and* contracts. They give the user permission to use the software (that is the license part) in exchange for variable consideration (that is the contract part). The user generally assents either by clicking "I Agree" when installing the software or by some other method. The GPL, some argue, is not a contract because no assent is required, and the user need not bind herself to anything.

case, is a copyright license known as the GNU GPL.[32] Under the GPL, Rockbox software developers agree that they will make their efforts available for download and modification; in exchange, people who make additions to the project or who want to distribute their work must also make their changes publicly available.[33]

The GPL has undergone serious academic scrutiny. Commentators do not agree on whether the agreement is a license or a contract, or whether that distinction has any meaning. Academics have discussed when an agreement is reached under the license; they debate what the remedy might be for breach of the contract (or failure to meet the terms of the license). But while there has been significant discussion surrounding the meaning of the GPL's terms, nearly every commentator has assumed that the license terms always have the same meaning. But why would this assumption follow? As we saw in Part II, identically worded private agreements need not be interpreted uniformly from party to party—and they rarely are.[34]

The GPL is one example of a contract that I refer to as a many-to-many contract.[35]

A. The GPL as a Many-to-Many Contract

The GNU GPL is a software license that has gotten quite a bit of recent press due to the success of the Linux operating system. It is also a many-to-many contract. In order to explain the structure of the contract, some background is needed.

Instead, the GPL has force because, should the user not abide by its terms, she will lack a license and will be liable for copyright infringement.

I use the words "contract" or "agreement" throughout to describe the GPL, but this contract versus license issue is mostly tangential to the issue of interpretation. Be it license or contract, the terms of the GPL must be construed. And similar standards of interpretation—and similar problems—crop up if the GPL is a license or a contract. *See infra* text accompanying note 35.

32. *GNU General Public License*, ROCKBOX, http://svn.rockbox.org/viewvc.cgi/trunk/docs/COPYING (last visited Jan. 20, 2012) [hereinafter GPL]. "GNU" is a recursive acronym that stands for "*G*N U is *N*ot *U*nix."

33. *Id.* This is a bit of an oversimplification of the license, as we shall see in Part III, but it is good enough for now.

34. In addition to the examples discussed in Parts II and III, differing circumstances may provide extrinsic evidence that would lead to different contract interpretations. Sure-Trip, Inc. v. Westinghouse Eng'g, 47 F.3d 526, 534 (2d Cir. 1995).

35. The word "contract" is not used here to take a side on the contract versus license debate I alluded to earlier; it is used, instead, because many-to-many contracts often span a wide range of private agreements, and a more precise designation ("privity of various kinds among various parties") would be unwieldy.

1. The Birth of the GPL: Coordinating Many Parties

Most software is purchased—either on CD or by download—in the form of object code, strings of ones and zeros that a computer interprets as a set of instructions.[36] Object code, however, is more difficult to modify than source code. While this series of ones and zeros is easily decoded by a computer, humans must expend considerable effort to decipher even the simplest programs that come in the form of object code. Computer programs, therefore, often start life as a series of human-readable instructions, typically written in the form of English-like commands.[37] These instructions are known as source code. Additional computer programs translate the human-readable source code into machine-readable object code.[38]

Unix is an operating system[39] developed by Bell Labs. It was used—at least in the 1970s—primarily in large research groups.[40] AT&T initially encouraged its users to play around with the innards of the operating system.[41] Technically, most research groups only purchased rights to the object code. But AT&T turned a blind eye when versions of the source code were "accidentally" exchanged among users.[42] The result was an explosion of Unix-based software.[43] One programmer's irritation about a feature of the operating system would result in commands and additions to the original source code. These additions were shared freely among programmers. Because most of the groups using Unix were in academic or laboratory-like settings, the ethic of sharing was ingrained.[44]

---

36. *See, e.g.*, JAMES P. COHOON & JACK W. DAVIDSON, C++ PROGRAM DESIGN 14-16 (3d ed. 2002).

37. *Id.* at 21.

38. *Id.*

39. An operating system, for those unfamiliar with the term, is the program that tells your computer how to turn itself on, how to accept input from the keyboard, and how to run other programs. Most people are familiar with Windows and Mac OS, which are examples of operating systems.

40. *See, e.g.*, Nick Moffitt, *Nick Moffitt's $7 History of Unix*, CRACK MONKEY, http://www.crackmonkey.org/unix.html (last visited Jan. 20, 2012); Eric Steven Raymond, *A Brief History of Hackerdom*, ERIC S. RAYMOND'S HOME PAGE, http://www.catb.org/~esr/writings/cathedral-bazaar/hacker-history/ (last updated Aug. 2, 2002).

41. Raymond, *supra* note 40.

42. *Id.* Eventually, this got them into trouble with FreeBSD, an operating system that lawyers can think of as similar to Linux. Real geeks, of course, know that Linux is superior. *See BSD Is Dying*, UNCYCLOPEDIA, http://uncyclopedia.wikia.com/wiki/BSD_is_Dying (last updated Jan. 31, 2011).

43. Raymond, *supra* note 40.

44. Richard Stallman, *The GNU Project*, GNU OPERATING SYSTEM, http://www.gnu.org/gnu/thegnuproject.html (last updated Sept. 20, 2011).

Around the time when copyrights in software began to be widely accepted, AT&T changed its mind about the availability of its Unix source code.[45] It began to crack down on sharing. Suddenly, users who solved problems and shared them with their colleagues were guilty of copyright infringement. Some users heralded the changes, noting that deserving programmers could recoup financial benefits for their investments.

Richard Stallman was not one of those users. Says Stallman, "I knew that at the end of my career, I would look back on years of building walls to divide people, and feel I had spent my life making the world a worse place."[46] In response, Richard Stallman started the GNU project. Along with Eben Moglen, he drafted the GNU GPL.[47] As Stallman says, the goal of the GPL is to keep free software free. Rather than restricting users' rights to modify, copy, and distribute the resulting code, the GPL allows users to do so at no charge.[48] The only restriction, however, is that any distribution of code that would infringe on the copyright of the original releaser must also be released under the GPL.

Some simple examples may clarify the license. Suppose that Aaron writes a program that prints simple text files and releases it under the GPL. Betty may then use Aaron's code under the license, free of charge.[49] She can give it to her friends, Celina and David. She can also modify the program so that it prints graphics, and she can distribute her modified version. The only restriction is that she must distribute her modified version—not only Aaron's original effort, but her additions—under the GPL as well.

David can further alter Aaron's and Betty's program by allowing it to print in color as well as black and white. The only catch? If David wishes to share his work, he too must distribute the entire resulting program under the GPL.[50]

The GPL, once attached to code, adheres through subsequent rounds of distribution and modification. In order to distribute the software without infringing on the original author's copyright, the distributor needs a license. The GPL itself says that "by modifying or

45. *Id.*; Raymond, *supra* note 40.

46. Stallman, *supra* note 44 (footnote omitted).

47. Free Software Found., Inc., *GNU General Public License, Version 2*, GNU OPERATING SYSTEM (June 1991), http://www.gnu.org/licenses/gpl-2.0.html.

48. GPL, *supra* note 32.

49. Under the GPL, Betty is still allowed to charge a transfer fee. *Id.*

50. *Id.*

distributing the Program (or any work based on the Program), you indicate your acceptance of this License."[51]

The goal was to keep software free from the restrictions that Stallman detested. Because successive modifications must be released under the GPL, successive generations will be able to rely on the code's continuing release under the appropriate license.[52] The license terms are somewhat complex, but can be summed up simply: You give what you get.

But the GPL has another effect besides freeing software from restrictions: because the parties need not negotiate a license for every new addition, large projects with multiple developers quickly become feasible. Everyone knows the terms of the release. The flip side of the terms of the GPL—the ease of distributing code and the certainty of the license—is that parties can coordinate their efforts with greater ease.

2. Many-to-Many Implications: The Rockbox Project

Legal academics, fueled in near-equal parts by developments in copyright law and the relative success of open-source software in the business world, have analyzed the GPL from nearly every contract perspective imaginable.[53] Most descriptions of the GPL in the legal literature proceed as I did above: we are exhorted to imagine party *A*

---

51. *Id.* Although the GPL precedes the statement I have quoted above with "[y]ou are not required to accept this License, since you have not signed it," there are two arguments that can be made. The first is that the GPL itself is the offer, and subsequent distribution is acceptance of the contract, with the license to distribute being adequate consideration. The other alternative is that the GPL is not a contract at all, but a license. I have referred to this issue before. *See supra* text accompanying note 31. If the GPL is treated as a license rather than a contract, the rules of construction are not quite as clear. *Compare* S.O.S., Inc. v. Payday, Inc., 886 F.2d 1081, 1088 (9th Cir. 1989) (requiring that license be interpreted against the licensee), *with* Bourne v. Walt Disney Co., 68 F.3d 621 (2d Cir. 1995) (using typical contract interpretation to construe license), *and* UNIFORM COMPUTER INFORMATION TRANSACTIONS ACT § 307(f) (1999) (same). But the reason I claimed that the contract versus license distinction was tangential is that this Part argues that treating drafter/draftee differently results in nonuniformity. If two parties intend to distribute code to each other, both must be licensor and licensee. Replacing "licensor" with "drafter" throughout this Part will result in similar confusion. There are some policy motivations that will not totally line up, but the changes you would need to make in the argument to make it align are basically trivial.

52. Technically, successive modifications need only be released under the GPL if the modified software is distributed to another party. You are free to modify the code for your own personal use, without releasing your efforts. GPL, *supra* note 32.

53. *See, e.g.*, Robert W. Gomulkiewicz, *De-bugging Open Source Software Licensing*, 64 U. PITT. L. REV. 75 (2002); Greg R. Vetter, *The Collaborative Integrity of Open-Source Software*, 2004 UTAH L. REV. 563; Wacha, *supra* note 31.

HeinOnline -- 86 Tul. L. Rev. 535 2011-2012

distributing to party *B*, party *B* modifying code and distributing it to party *C*, and so forth.[54]

These simple interactions are useful when the goal is to explain how the license works. Yet the simplicity associated with breaking transactions under the GPL into bite-sized, two-party pieces hides complexity. If we want to understand what Stallman intended by the license, the description used above is adequate: identify two contracting parties and discuss their obligations to each other under the GPL.

But two-party interactions fail to capture the complexity of the GPL's contractual structure. Because each programmer retains copyright in her code under the GPL, the number of license agreements needed under a typical project released under the GPL is dramatically larger than the number of license agreements involved with typical software licensing.[55] More importantly, because the license agreements cover overlapping rights and software, the benefits and burdens imposed by the GPL on one pair of parties, *A* and *B*, cannot be construed independently of the benefits and burdens associated with another pair, *B* and *C*.[56]

An analysis of a software program released under the GPL, known as the Rockbox project, illustrates these points.[57] Rockbox is "firmware"[58] that replaces the operating system on the "Jukebox" series of portable music players produced by Archos.[59] The Rockbox project is ideal for an initial study of the complexity in GPL-based projects because it is relatively self-contained: any program included in the Rockbox project must be able to run on the Archos Jukebox's particularized hardware. It is therefore *technologically* difficult to "borrow" code from other sources, even if the GPL makes it *legally*

---

54. David McGowan, *Legal Implications of Open-Source Software*, 2001 U. ILL. L. REV. 241, 257-58.

55. *See infra* Figure 1 and accompanying text.

56. *See infra* Part III.B.

57. ROCKBOX, *supra* note 26.

58. "Firmware" is much like software; the term is often used to refer to a program that makes a particular device run.

59. ROCKBOX, *supra* note 26. An operating system on a music player is the piece of software that shows what music files—usually saved in MP3 format—are saved on the player and provides the necessary tools to facilitate playback. The original firmware supplied by Archos had—to put it gently—more bugs than features. The Jukebox, however, was an inexpensive yet elegant piece of hardware. The Rockbox developers decided to produce an operating system worthy of the hardware by reversing the bug to feature ratio. In addition to removing a large number of bugs, the Rockbox project has provided an operating system that can operate in multiple languages, can read file names out loud for the hearing impaired, and can provide games and a personal organizer. *See supra* notes 25-30 and accompanying text.

HeinOnline -- 86 Tul. L. Rev. 536 2011-2012

trivial.[60] In terms of party structure, then, the Rockbox project is relatively simple compared to many other projects released under the GPL.[61]

At the end of 2002, Rockbox had twenty-five developers working on the project.[62] Each of those developers retained copyright in his or her work and allowed others to copy, modify, and distribute that work pursuant to the GPL. Because copying, distributing, and running computer code would be an act of infringement in the absence of a license, every time a developer uses code that was created by another programmer, the developer must have a license from that programmer.

Let us suppose that Björn Stenberg first established the project.[63] Thereafter, individuals wishing to make modifications to Stenberg's original work would need Stenberg's permission, lest they violate copyright.[64] So when Linus Nielsen Feltzing added, for instance, the

60. Source code is actually relatively portable. However, in comparison with (for instance) personal computers, the Archos Jukebox has limited processing capabilities and minimal system memory. Programs written for personal computers may demand more resources than are available on the Rockbox. As it turns out, Rockbox borrows very little from other programs, but it does borrow. Sorting procedures and a screensaver, at a minimum, appear to be borrowed from other open-source projects.

61. Even though software released by, for instance, Microsoft, requires thousands more developers, the party structure is much simpler. Microsoft developers do not own the copyright to their work, instead creating on a work-for-hire basis. Thus, when you install Microsoft Office, you only need a license from a single copyright holder: Microsoft.

62. This is, by necessity, an underestimate. The number of developers was estimated by searching through the Rockbox source code for lines containing the word "Copyright." This line usually contained the author's name and a date of copyright. The name and date were extracted to a file and sorted by date. It is assumed that any name with a copyright date of 2002 was a developer at least in 2002.

63. I have not yet found a compressed history of the Rockbox project, but Björn Stenberg answered questions about the project when it was first announced on Slashdot, a news and conversation site "for nerds." *See* Björn Stenberg, Comment to *Rockbox Replaces Archos Firmware*, SLASHDOT (June 4, 2002), http://slashdot.org/articles/02/06/04/0018259.shtml. Furthermore, in determining the name of the Rockbox project, Björn Stenberg single-handedly nixed the democratically voted "Orpheus," referring to himself as a "dictator" in so doing. *See* Björn Stenberg, *Rockbox Mail Archive: Name Change*, ROCKBOX (Jan. 16, 2002), http://www.rockbox.org/mail/archive/rockbox-archive-2002-01/0134.shtml.

I think I am fairly safe in attributing first authorship to him.

64. Modifications of the Rockbox type are probably derivative works, at least in the United States. At a first glance, it seems that the same is true in the European Union. It is possible that some of the developers on the Rockbox project would not need a license in order to release their modification. This will be particularly true if the modification stands alone. I have not examined the code in great detail to determine if this is the case, so it is possible that this diagram overestimates the complexity of the interactions. Because I also think that there are ways in which I am underestimating this complexity, see *infra* text accompanying note 67. I do not think this is a large worry.

ability for Rockbox to interact with a computer's USB port,[65] he licensed Stenberg's original work. If Uwe Freese decided to modify Rockbox so that the music player would shut off after a certain amount of idle time,[66] he would need a license from both Feltzing and Stenberg. Likewise, if Feltzing wishes to use Freese's code, he would also need a license. We can imagine this proceeding so on and on with all twenty-five developers. In the end, what we get is a set of licenses, spreading like a web among all twenty-five developers. Figure 1 shows this web of licenses.

The number of license arrangements necessary grows geometrically with the number of parties. In the one-to-many context, an insurer that contracts with twenty-four different insured's would have twenty-four different contracts. By contrast, if Björn Stenberg works with twenty-four different software developers, there will be as many software arrangements between all twenty-five parties as there are lines in Figure 1. For purposes of quantifying, the total number of licenses is $24 + 23 + 22 + 21 + \ldots + 1 = 300$.

Going beyond that first year, the license entanglement only becomes more complex than depicted. First, these are only the developers associated with the project in its first year. In 2003, eight new developers joined the project. They needed licenses from the original 2002 Rockbox developers—and as their code became incorporated into the project, the original developers would need licenses from them as well, resulting in $32 + 31 + \ldots + 1 = 528$ total agreements. In 2004, another four developers joined the Rockbox project, bringing the total number of agreements up to a diabolical 666. And this number leaves off several pieces of code that were imported into Rockbox from other works released under the GPL. Finally, the figure only includes people who have modified the source code. Mere downloaders must also have a license from every developer, but they are not represented.[67]

I chose the infant Rockbox project because of its relative simplicity. Compared to other programs released under the GPL,

65. *See* Linus Nielsen Feltzing, *usb.c*, ROCKBOX PROJECT VERSION 2.4 (on file with author).

66. *See* Uwe Freese, *powrmgmt.c*, ROCKBOX PROJECT VERSION 2.4 (on file with author).

67. For instance, pursuant to the GPL, and in order to preserve the data for this Part, I retained a copy of the Rockbox source code on my personal Web site. In order to distribute this code to the world at large, I must also license the code from all thirty-eight Rockbox developers. If you choose to download the code and then send it to a friend, you will also need a license from all the developers.

HeinOnline -- 86 Tul. L. Rev. 538 2011-2012

Rockbox has a very small number of contributors. Linux, the most famous open-source project, coordinates the activities of several thousand programmers in applications development alone.[68] Had I attempted to represent Linux as I did in Figure 1, the graphic would have required a sheet of paper the size of Kansas, liberally covered with millions of lines.[69]

It is, of course, not the *number* of licenses that matter; after all, any insurance company worth its salt sells far more than six hundred, or even three thousand, contracts. It is the structure of interaction between the licensees that matters. As the next Parts explain, this interaction may mean that uniformity of interpretation, at least in the many-to-many GPL context, is vital for the terms of the contract to make sense.

B. Interpreting the Many-to-Many GPL

Ambiguities in the GPL have been discussed in detail elsewhere.[70] For instance, Robert Gomulkiewicz claims that the GPL is "buggy—out of date, misapplied, misunderstood and hopelessly confusing."[71] Gomulkiewicz identifies several places where misunderstandings could arise.[72] For instance, the GPL explicitly says that running a program is not covered because the license's scope extends only to

---

68. Bert J. Dempsey et al., *A Quantitative Profile of a Community of Open Source Linux Developers*, IBIBLIO (Oct. 6, 1999), http://www.ibiblio.org/osrt/develpro.html. Note that this study took place far before Linux's recent major growth.

69. Technically, regular 8½ x 11 paper could be used, but at some point the lines would be indistinguishable by the human eye.

70. *See generally* Gomulkiewicz, *supra* note 53 (discussing ambiguities of the GPL).

71. *Id.* at 76 (footnotes omitted). Because Gomulkiewicz identified these flaws, several newer versions of the GPL have been released. I do not use it for two simple reasons. One, the infant Rockbox project relied on the earlier version of the GPL. Two, the point here is not to illustrate an actual scenario that could occur—in truth, if this sort of problem ever comes into court it will undoubtedly involve a much more complex issue than I discuss here and will be unlikely to involve two private individuals arguing about code that runs MP3 players—but instead to demonstrate how, in the many-to-many context, uniformity is important.

72. Nor is he alone. *See* Peter Brown, *Legal Issues in the Open Source Community*, 780 PRACTISING L. INST. 309, 319 (2004) (noting that GPL terms are ambiguous); David S. Evans & Bernard J. Reddy, *Government Preferences for Promoting Open-Source Software: A Solution in Search of a Problem*, 9 MICH. TELECOMM. & TECH. L. REV. 313, 340 (2003) (stating that there is ambiguity about when terms of GPL come into play); Christian H. Nadan, *Open Source Licensing: Virus or Virtue?*, 10 TEX. INTELL. PROP. L.J. 349, 360 n.48 (2002) (claiming that the GPL is ambiguous with respect to linking to a GPL-based program); Wacha, *supra* note 31, at 486 (explaining why ambiguity regarding derivative work status is not fatal to GPL); Brian W. Carver, Note, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443, 458 n.84 (2005) (noting that GPL's definition of a "derivative work" is ambiguous).

copying, distribution, and modification.[73] And yet running a program without a license would infringe copyright.[74] This sort of ambiguity can be easily resolved, as generally a license to run the program would be implied by party behavior.[75] Nonetheless, this approach is not without worries.

This Part demonstrates how apparent ambiguities in the GPL could be exploited under the rule of contract construction that disfavors the person who supplies the language.

Early on, I should note that while I attribute irrational and silly behavior to some of the Rockbox developers to illustrate the problem, it is highly unlikely that any of them would ever act in this fashion. Software programmers working on free projects are rarely litigious sorts and would be more likely to work their differences out amongst themselves. And it is even less likely that parties would exploit any potential ambiguity with respect to "running" the program as I describe here. It is far more likely that outside parties may attempt to profit from the Rockbox project's good work by exploiting an ambiguity as to whether dynamically linked programs would need to be covered by the GPL. If the GPL is ambiguous in this regard, companies who want to save on software development costs may attempt to use code released under the GPL without distributing their modifications in source code format under the GPL. This evasion of the GPL's reciprocal burdens may very well result in lawsuits to enforce the copyright.

Just as in the one-to-one context, under the Restatement of Contracts, ambiguities in the GPL should be interpreted against the party "who supplies the words."[76] This does not just affect Richard Stallman, the drafter; it would also count against developers who started a project and then chose to release it under the GPL. In other words, the rule would be construed against Björn Stenberg.

This rule, according to the Restatement, is "often invoked in cases of standardized contracts."[77] Without a doubt, the GPL is a standardized contract. While the GPL mentions that authors may renegotiate different licenses,[78] this is both rare and unlikely. Because

73. GPL, *supra* note 32.

74. Gomulkiewicz, *supra* note 53, at 84-85.

75. *Id.* at 85.

76. RESTATEMENT (SECOND) OF CONTRACTS § 206 (1981).

77. *Id.* § 206 cmt. a.

78. GPL, *supra* note 32 ("If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission.").

HeinOnline -- 86 Tul. L. Rev. 540 2011-2012

each author retains her copyright, an individual seeking to relicense the 2002 Rockbox project would need to negotiate licenses with all twenty-four developers. Furthermore, while some projects advertise their willingness to release non-GPL versions of their projects,[79] many developers—including the Rockbox developers—are vociferous proponents of the GPL, who are likely unwilling to release the code under a different license.[80]

Bargaining power is often disproportionate. Some contributors to GPL projects are large corporations—such as IBM, Sun Microsystems, and Intel.[81] Even if the only contributors are individuals, free of bureaucratic tendencies, who are willing to renegotiate in the abstract, the transaction costs of attempting to renegotiate a license from each of the developers would be staggering.[82] One of the 2004 Rockbox authors describes himself[83] as "Pengxuan (Isaac)" with a Yahoo! e-mail account.[84] Another is listed

79. MySQL, a database program, offers both a commercial and an open-source license. *Legal Policies*, MYSQL, http://www.mysql.com/about/legal/ (last visited Jan. 20, 2012). Trolltech also produces a code development framework available under both commercial and open-source licenses. *Licensing Overview—Trolltech Products*, TROLLTECH (Apr. 15, 2006), http://web.archive.org/web/20060415012915/http://www.trolltech.com/products/licensing.html (accessible only as archival material through The Wayback Machine).

80. See, for instance, the interaction with a user named Bluechip. BlueChip, Comment to *Rockbox Mail Archive: Re: Writing Plug-Ins*, ROCKBOX (Jan. 10, 2005), http://www.rockbox.org/mail/archive/rockbox-archive-2005-01/0281.shtml; Jerry Van Baren, Comment to *Rockbox Mail Archive: Re: Writing Plug-Ins*, ROCKBOX (Jan. 12, 2005), http://www.rockbox.org/mail/archive/rockbox-archive-2005-01/0344.shtml.

81. Steven J. Vaughan-Nichols, *United Linux: One for All, and All for One*, PRAC. TECH. (May 30, 2002), http://web.archive.org/web/20060214002839/http://www.practical-tech.com/infrastructure/i05302002.htm (accessible only as archival material through The Wayback Machine).

82. Molly Shaffer Van Houweling, *The New Servitudes*, 96 GEO. L.J. 885, 939-41 (2008). For a somewhat related critique, see Rod Dixon, *When Efforts To Conceal May Actually Reveal: Whether First Amendment Protection of Encryption Source Code and the Open Source Movement Support Re-Drawing the Constitutional Line Between the First Amendment and Copyright*, 1 COLUM. SCI. & TECH. L. REV. 3 (2000), http://www.stlr.org/cite.cgi?volume=1&article=3 ("[O]ne obvious, practical, and destabilizing effect of an open[-]source project based on GPLs is the potential for the project to implacably acquire copyright interests in all the works for which the original source code is based."). Some of Dixon's concern seems to be based on a misreading of the license; the GPL does not require that copyright be "aggregate[d] in one original 'author,'" *id.*, only that the original author (as well as the rest of the world) be given a nonexclusive license to later works. GPL, *supra* note 32.

83. Or herself, although this seems less likely with the parenthetical "Isaac."

84. Pengxuan, *calculator.h*, ROCKBOX PROJECT VERSION 2.4 (on file with author). On both April 11, 2005, and December 20, 2005, I sent "Pengxuan" an e-mail asking about the Rockbox project. He did not respond to either e-mail.

simply as "wavey@wavey.org" with no name.[85] Other authors list names but no contact information. The difficulty of simply finding some of these people—or worse yet, their heirs—would be immense.

This problem can be illustrated with another effort to change a license for a software project where the copyright was held by a large number of individuals. Mozilla—the successor to the Netscape web browser[86] and a precursor to the well-known Firefox browser—was originally licensed under the Mozilla Public License.[87] For a number of reasons, several prominent members of the Mozilla project decided to release the project under the GPL.[88] A general call was made to anyone who had contributed code to the project—at the time, some 300-odd developers—to authorize the new license.[89] Months later, they issued pleas to the geek community to please help them track down the remaining programmers.[90] The Mozilla project is currently released under the GPL, but the relicensing project still lists several developers as unfound.[91] Did they find these individuals, but fail to update the Web page? Did they excise the code contributed by the missing people? Or did they just shrug their shoulders and let it go?[92]

---

85. wavey@wavey.org, *panic.h*, ROCKBOX PROJECT VERSION 2.4 (on file with author). On both April 11, 2005, and December 20, 2005, I sent "wavey" an e-mail asking about the Rockbox project. Surprisingly, wavey did reply to my second e-mail. E-mail from Stuart Martin to author (Dec. 20, 2005, 7:42 PM) (on file with author).

86. *Mozilla*, WIKIPEDIA, http://en.wikipedia.org/wiki/Mozilla#History_of_Mozilla (last updated Jan. 8, 2012).

87. *Mozilla Public License Version 1.1*, MOZILLA, http://www.mozilla.org/MPL/MPL-1.1.html (last visited Jan. 20, 2012).

88. The Mozilla Public License was deemed to be "incompatible" with the GPL. In order to use available code that would simplify the project, Mozilla actually needed to be released under a different license. To be perfectly accurate, they did not relicense Mozilla under the GPL. Instead, they licensed it under the Lesser General Public License.

89. *Mozilla Relicensing FAQ*, MOZILLA, http://www.mozilla.org/MPL/relicensing-faq.html (last updated Aug. 14, 2007).

90. *Have You Seen These Hackers?*, MOZILLA, http://www.mozilla.org/MPL/missing.html (last updated Jan. 20, 2012).

91. *Id.*

92. This reality has been recognized by the Linux community.

> All code merged into the mainline kernel retains its original ownership; as a result, the kernel now has thousands of owners.
>
> One implication of this ownership structure is that any attempt to change the licensing of the kernel is doomed to almost certain failure. There are few practical scenarios where the agreement of all copyright holders could be obtained (or their code removed from the kernel).

Jonathan Corbet, *A Guide to the Kernel Development Process*, LINUX FOUND. (May 12, 2011, 7:22 PM), http://www.linuxfoundation.org/content/1-guide-kernel-development-process. For that same reason, the developers have recognized that anonymous contributions pose serious dangers.

I do not know the answer; nonetheless, the sheer magnitude of the effort involved in relicensing Mozilla suggests that a person of average resources would be unable to negotiate a license for a project released under the GPL.[93] The end result is that while each developer, individually, may be small potatoes, collectively, they are a force to be reckoned with.

When it comes down to it, the GPL is a potentially ambiguous standardized license, chosen by parties that effectively have disproportionate bargaining power. Most projects released under the GPL are difficult to relicense. A suit brought against makers of GPL software claimed that the GPL was a contract of adhesion.[94] Several others have noted that the GPL, and similar licenses, would be construed against the party that drafted (or decided to use) it.[95]

In fact, the case for construing the GPL against its providers may seem particularly strong. The party that provides the language here binds any subsequent party to release under the GPL. The language is not negotiated; even parties that generally like the substance of the

---

> It is imperative that all code contributed to the kernel be legitimately free software. For that reason, code from anonymous (or pseudonymous) contributors will not be accepted. All contributors are required to "sign off" on their code, stating that the code can be distributed with the kernel under the GPL.

Id.

93. There may well be exceptions; take, for instance, the Reiser Filesystem, a set of tools that create, maintain, and repair a "journaling filesystem." The copy notice for that file states:

> Further licensing options are available for commercial and/or other interests directly from Hans Reiser: reiser@namesys.com. If you interpret the GPL as not allowing those additional licensing options, you read it wrongly, and Richard Stallman agrees with me, when carefully read you can see that those restrictions on additional terms do not apply to the owner of the copyright, and my interpretation of this shall govern for this license.

Hans Reiser et al., *Copyright for "reiser4progs" Source Package in Maverick*, UBUNTU, https://launchpad.net/ubuntu/maverick/+source/reiser4progs/+copyright (last visited Jan. 20, 2012).

94. Wallace v. Free Software Found. Inc., No. 1:05CV0618-JDT-TAB, 2005 WL 3239208, at *5 (S.D. Ind. Nov. 28, 2005). Daniel Wallace claimed, among other things, that the GPL is a boilerplate contract that authors have no choice but to accept and that the terms of the GPL violate antitrust law. His argument was rejected by the United States Court of Appeals for the Seventh Circuit. Wallace v. IBM Corp., 467 F.3d 1104 (7th Cir. 2006). For a lengthier explanation of both his argument and a potential response, see Heidi S. Bond, *What's So Great About Nothing? The GNU General Public License and the Zero-Price-Fixing Problem*, 104 MICH. L. REV. 547 (2005).

95. Gomulkiewicz, *supra* note 53, at 80 n.40; Timothy Hadley, *Creative Commons Licensing: How Much, for How Long, Who, and Why?*, MATH CLASS FOR POETS (Mar. 12, 2003), http://www.tph-lex.com/old/mcfp/2003/03/creative-common.html. Intellectual property law may have more to say about this, but that is outside the scope of this Article.

GPL may be unable to negotiate similar licenses from all releasing parties. In fact, because the GPL does not allow you to add or modify terms in the license, in some circumstances, it may effectively bar relicensing unless everyone who contributed code agrees.[96]

But what does this mean? First, we must determine which party "supplies" the language. It is not as simple as the one-to-many case, where one person supplies the language and many others accept it. This is because, unlike the one-to-many case, two-party contract pairs are difficult to treat separately.

On rare occasions, you may be able to treat two-party pairs analytically. Suppose that Björn Stenberg, one of the original 2002 developers, was the first person to release code under the GPL. Because Stenberg had his choice of a number of licenses, including licenses that he could write himself that resolved ambiguities in the GPL in a more satisfactory manner, it would be hard to imagine that Stenberg was not the party that supplied the language of the license. And so it seems perfectly reasonable, when examining the two-party pairing of Stenberg with Felix Arends, to imagine ambiguities between Stenberg and Arends being resolved against Stenberg.

But if Stenberg is the sole original supplier of language, pairings that do not contain Stenberg might not be construed in a similar fashion. Take the agreement between Arends and Uwe Freese. Neither Arends nor Freese really had any control over the language of the license. By the terms of the GPL, if they wanted to make additions to the Rockbox project, they had to release their work under the GPL. Thus, many of the rationales typically associated with the general rule of contract construction—the supplier of language is in a better position to be aware of and correct misapprehensions, and the court wishes to motivate him to evaluate ambiguities—apply only to Stenberg, who is not a party to the Arends/Freese pairing.

Of course, this does not pose an inherent problem with respect to contract interpretation. In the absence of a party to construe language

96. Envision the following: Archos, the company that produced the hardware on which the Rockbox project runs, wants to negotiate a license with the Rockbox project. While the company has no problem, generally, with the terms of the GPL, they want to add a start-up graphic that refers to the Archos Web site. They also want to add a term to the license preventing anyone from removing that from the start-up process. Now imagine that Archos can get thirty-seven of the thirty-eight Rockbox developers to license under those terms. If recalcitrant programmer number thirty-eight refuses to release under the modified almost-GPL, and if the material contributed by that programmer is heavily integrated into the project such that it cannot be removed easily, that one programmer can prevent Archos from releasing under that license, even though the vast majority of his compatriots disagree.

HeinOnline -- 86 Tul. L. Rev. 544 2011-2012

against, ambiguities in contract language usually do not render the whole totally indeterminate.[97] In this case, contract interpretation would probably proceed as in any contract where one party does not supply the language.

The only catch—and it seems at first a minor one—is that the presence of a rule of contract construction between Stenberg and Arends that is absent between Arends and Freese may mean that the license between Stenberg and Arends may be interpreted differently than the license between Arends and Freese.

Well and good; the issue seems indistinguishable from the one-to-many context discussed above. We interpret the meaning of the contract based on who supplied the language. Occasionally this means that identical contracts are interpreted differently. Yet one-to-many contracts need not be interpreted consistently among contract pairs; aside from a subtle cognitive dissonance, this should be dismissed with an airy hand motion and murmurs about hobgoblins and foolish consistency.

Or should it? In the insurance context discussed earlier, Jane, when facing Best Insurance's protests that the term had been defined in its suit versus John, asked, "What the devil has *John* got to do with *me*?" In the one-to-many context, the contract pairings overlapped at *one* party only—the original provider of the language. In the many-to-many context, more parties overlap.

Arends, in discussing his license with Freese, cannot bluster that his license with Stenberg is irrelevant for two reasons. First, Arends must extend a license to Freese because it is required by his license with Stenberg. Second, Arends, Freese, and Stenberg's work are collected as a whole, and released to the public with a single text file accompanying it, containing what are purported to be the license terms.[98] If we interpreted the license according to initial party relationships, someone wishing to build on the entire work who was curious about the scope of her license would need to take into account who wrote what pieces of code and whether that person made the original choice to release under the GPL. Despite the unitary feel of the project, the single copyright and distribution notice, and the

97. And if they do render it wholly ambiguous, the contract is usually considered rescinded for lack of mutual understanding. *See* Raffles v. Wichelhaus, (1864) 2 Hurl. & C. 906, 159 Eng. Rep. 375 (Exch.).

98. *See* ROCKBOX PROJECT VERSION 2.4 (on file with author). Every file in the Rockbox project contains a copyright notice with a date, the authors' names, and a statement to the effect that the code contributed is released under the GPL, a copy of which can be found under the file "COPYING" in the archive.

HeinOnline -- 86 Tul. L. Rev. 545 2011-2012

uniform wording of the license under which it was released, the project would be effectively released under multiple licenses. The scope of each license could be discerned, not by the words used in the license, but only by whether the developer in question was part of the initial choice to release the project under the GPL.

I have stated this objection in very abstract terms; we can quickly make it more concrete by examining Gomulkiewicz's objection that the GPL's coverage of "running" is ambiguous.[99] Because the license is ambiguous, imagine a person—let us call him Jeremy—starting a new software project, which I will call CodeOnly.

Says Jeremy, "I'm releasing CodeOnly under the GPL. But the GPL is ambiguous about whether you have the right to run the program. Usually, of course, people would imply a license to run the program in the face of such ambiguity. But when I release my code under the GPL, I intend only to give potential users the rights to copy, modify, and distribute the code under the GPL. I want to make it perfectly clear that I interpret the GPL to exclude the right to run; if you'd like to run the program, you can pay me for an additional license."

There is nothing legally wrong with this.[100] Naturally, Jeremy's interpretation of the GPL does not mean the same thing as the GPL's original author, Richard Stallman. But if Jeremy goes out of his way to clarify his intent, and if he makes sure that everyone who downloads his code is aware that his interpretation is different and agrees to it, this difference should not pose a problem. It is a stupid way to interpret the license, but it is not inconsistent with the license wording, and we cannot imply a term in the teeth of a specific clarification to the contrary between the parties.[101] The fact that Stallman interprets the GPL differently is essentially irrelevant. After all, what the devil does Richard Stallman have to do with Jeremy?

---

99. *See supra* note 72 and accompanying text.

100. The result is ridiculous from a contract interpretation perspective. There would then be little reason to download his project, except perhaps to examine the code for brilliant insights. Nevertheless, parties are allowed to make ridiculous contracts. As long as there is a meeting of the minds, courts will not usually second-guess ridiculousness.

101. Essentially, the reason courts will not do this is that Jeremy's repeated insistence that users cannot run his software constitutes extrinsic evidence of his interpretation of the GPL. While some users may have different interpretations of the GPL, they are probably acquiescing in his interpretation by downloading and running his software. One can imagine "battle of the forms" like situations where an individual insists his license means something else, but his insistence would be unlikely to sway a court.

HeinOnline -- 86 Tul. L. Rev. 546 2011-2012

But what if we import Jeremy's ridiculousness into the many-to-many Rockbox project?[102] Stenberg, of course, releases Rockbox under the GPL. He is silent as to whether the license allows running. Along comes Arends, who wants to modify Stenberg's code. Stenberg, by releasing Rockbox under the GPL, has given Arends license to do this, as long as Arends complies with the terms of the GPL, which Arends claims to do willingly. So far, so good.

But imagine that when Arends distributes the resulting code to Freese, he adds Jeremy's disclaimer. "Dear Uwe," he types in the accompanying e-mail, "I am sending you a copy of Stenberg's code, with my modifications. The whole is released under the GPL. But I wanted to clarify one ambiguous portion of the GPL with respect to my modifications. I am specifically withholding the right to run the program. I would be delighted to give you a license to run my portion of the code for a small fee."

Outraged, Freese contacts Stenberg. Stenberg can deal with this in a number of ways. "Fine," he can say to Arends, "my license to you also disallows running the program." Because Stenberg was silent about running in the beginning, as the supplier of the license, ambiguities would be interpreted against him. "Look," says Stenberg to Arends, "be reasonable. I gave you the ability to run the program; you should give the same thing to Freese." To which Arends responds: "What the devil does *Freese* have to do with my license with you?"

And here is the devil in the details. The crux of the distinction between one-to-many and many-to-many contracts is that Stenberg's license to Arends has *everything* to do with his license to Freese. Stenberg will argue that the GPL's requirement that derivative works be licensed "under the terms of this license" means that Arends must release his modifications not only under a license with identical

102. I do not, of course, claim that the Rockbox participants I have named—or ones that I have not named—are such litigious sorts that they would erupt into legal arguments at the drop of a hat. The first thing to note is that legal challenges to the GPL will almost certainly result from people outside the community who distribute GPL-based code without complying with the terms of the license. I have not used the more persuasive ambiguity in the GPL here—the fact that it is not clear if you have to distribute code under the GPL if your contribution is dynamically linked to GPL software. I chose to discuss running the program for purposes of explaining the problem. I suspect that many people will find the distinction between dynamic and static linking of source code confusing, but that most will understand what it means to run a program. As software released under the GPL increases in importance, the chances of someone using GPL code in a fashion that is likely to outrage GPL users, but could arguably be allowed under the license, increases. In fact, the Santa Cruz Operation group has already done this. *See Frequently Asked Questions*, SCO, http://www.sco.com/scosource/linuxlicensefaq.html (last visited Jan. 20, 2012).

language, but also under a license with identical *interpretation*.[103] After all, the intent of the license was to guarantee that you gave what you got. In other words, the terms of the license dictate not only what you receive, but what you have to pass on to others in turn.

The outcome of applying the typical rule of construction to Arends is that the two rules that we discussed in the one-to-many context—namely, that the specific trumped the general, and that language should be construed against whomever supplied it—can cause serious problems. If Arends's interpretation of the language allows every potential ambiguity in the language to drive a wedge between the parties, it would inhibit the one thing that the GPL was intended to facilitate: the free exchange of software. Instead of the interconnected black lines of Figure 1, every line might potentially be interpreted differently,[104] depending on what the parties had specifically communicated to each other, potential ambiguities, and who originally released the software under the GPL.

This did not bother us in the insurance context, but the insurance context does not depend upon later transfers. You cannot change your house without changing your insurance; you cannot sell your car insurance to your neighbor or let a friend borrow your life insurance when he embarks on a dangerous journey. Your insurance policy has nothing to do with anyone else's, save for the fact that it may be issued by the same entity.[105]

Rockbox, and other GPL-based projects, differ. If everyone's license meant different things, I could not place the Rockbox project source code on my Web site, as I have done, and be certain of my rights. You, the reader, could not safely download the code from my site and run it without knowing about relationships and communications between people you do not know and do not care to meet. You would have to see who wrote the part of the program that played back music files and make sure that they allowed you to run it. You would need to verify that all thirty-eight people agreed with a particular permissive interpretation of the GPL in order to rest free. And for larger projects, where even more people contributed under the GPL, finding out what your license meant would be nearly as much work as trying to relicense the code under a separate license.

---

103. GPL, *supra* note 32.

104. Assuming a large number of reasonable interpretations of the license.

105. To some extent, your insurer—or your mortgage provider—can resell your insurance contract or your mortgage.

Furthermore, as some of the ambiguities identified by others became ambiguous only in light of technological advances *after* the license had been written,[106] the authors of the contract would need to be psychic as well as careful.

Allowing differing interpretations of the GPL to flourish would destroy the project. Instead of being able to download a program, with some idea that it was protected by one copyright license, potential variations in every license would need to be explored and understood. The resulting chaos could essentially destroy projects released under the GPL. And that is something that no party intended at all.[107]

Of course, while intent matters in contract interpretation, the rule of *contra proferentem* is not wholly motivated by the intent of the parties and is related to the policy goal of encouraging contract drafters to avoid ambiguity.[108] But here, too, the rationale breaks down in the context of the GPL. The goal of construing ambiguities against the drafter is to place the burden of avoiding ambiguity on the drafter. But the burden of the GPL's ambiguity would not fall entirely—or even primarily—on Stenberg. If Arends insists that the GPL does not allow others to run his code, it is Freese, the other Rockbox developers, and the thousands of Rockbox users who will pay the price.

In the many-to-many context, when private agreements provide public benefits, construing ambiguous terms against the provider of the language would not place the burden of unclear drafting on the shoulders of the person who chose the language.

IV. CONSTRUING MANY-TO-MANY CONTRACTS: WHY UNIFORM INTERPRETATION IS REQUIRED

It is not enough, of course, to decry a canon of construction by saying that it would lead to chaos among the parties. After all, if the parties foster chaos through poor drafting, it is not the job of contract

106. In particular, the linking of some files in a particular way is ambiguously covered by the GPL, but "static linking," the source of the ambiguity, did not exist when the GPL was drafted. Gomulkiewicz, *supra* note 53.

107. In this case, the GPL specifies its purpose:

> Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

See GPL, *supra* note 32.

108. K.M. Sharma, *From "Sanctity" to "Fairness": An Uneasy Transition in the Law of Contracts?*, 18 N.Y.L. SCH. J. INT'L & COMP. L. 95 (1999).

law to save them. But the many-to-many contract differs from the one-to-many contract in two ways. First, both parties *intend* a uniform interpretation. And second, the burden of ambiguity cannot be easily placed upon the person who drafted the language. Another example of a many-to-many contract may clarify and demonstrate how the canons of contract interpretation come into play in the many-to-many contract.

The last thing that comes to mind when discussing licenses governing the distribution of free software may be bond indenture contracts between large corporations and savvy investors.[109] But that is where we find the solution to the problem posed in Part III.

Start with a problem that faces many bond investors. The interest rate paid on corporate bonds is typically related to the health of the entity underwriting the bonds.[110] It is a simple matter of risk aversion; a company that teeters on the brink of Chapter 11 bankruptcy is more likely to renege on its obligations, and so will be assessed significantly higher interest rates than a corporation that generates mountain-high profits.[111]

Nothing guarantees that the risk the investors bargained for in the beginning—the risk that the company will be unable to repay its obligations—will remain the same, or even that the company will attempt to minimize risks to bond holders. But investors, as a general rule, try to make sure that companies do not, by means of somewhat underhanded transfers of important assets, transform what was once a stodgy but respectable low-risk, low-return investment into the worst possible investment vehicle: the high-risk, low-return bond. In the absence of fiduciary duties to bond holders, these protections are primarily contractual.[112] These agreements typically restrict the issuer's ability to increase the corporate debt load, but more importantly often dictate whether—and how—debt can be sold or assigned to a different entity.[113]

Multibillion dollar lawsuits arise when the parties disagree on exactly what sorts of transfers the contract restricts. Some parts of the bond indenture agreement—for instance, terms of repayment and interest rate—are usually dickered.[114] Transfer agreements are also

---

109. Bond indentures are often the last thing to come to mind, no matter the topic of discussion.

110. *See* AM. BAR FOUND., COMMENTARIES ON INDENTURES 1-2 (1971).

111. *Id.*

112. *Id.* at 2.

113. Metro. Life Ins. Co. v. RJR Nabisco, Inc., 716 F. Supp. 1504, 1512 (S.D.N.Y. 1989).

114. *See, e.g.*, AM. BAR FOUND., *supra* note 110.

dickered to the extent that the parties may decide what corporate actions are prohibited or must be approved by a trustee.[115] However, the parties involved rarely negotiate the precise verbiage of the transfer term.[116] Instead, a party might choose one possible wording from several boilerplate agreements. Those words would then be slapped into the bond agreement without modification.[117] It is important to note that both parties here are often relatively powerful. The investor that pumps money into a bond issue may be able to choose the agreement that he wishes.

Following the simple model of one-to-many contract interpretation might lead one to conclude that the party that chooses the boilerplate should, all things being equal, have ambiguities construed against him.[118] But like the Rockbox case, this interpretation would devolve into chaos.

Consider a simple circumstance that could cause confusion. Suppose Alex negotiates a bond indenture agreement with IndebtedCorp. Because Alex has a preference as to the antidilution terms, he asks IndebtedCorp to use a particular boilerplate form. They agree to use the language that Alex supplies.

But Alex does not want to fund the entire issue. So IndebtedCorp finds a second investor: Andrea. Because Andrea's bonds will be part of the same issue as Alex's, IndebtedCorp uses the agreement that Alex supplied and refuses to negotiate it.

In Alex's bond issue, Alex supplied the language. In Andrea's bond issue, IndebtedCorp supplied the language. According to the normal rules of construction, ambiguities in the language might be construed against Alex, but in favor of Andrea. After all, we are dealing with boilerplate agreements here, and the inference of construing the language against the supplying party should be particularly strong.

What then of James, the inveterate bond investor? James is a connoisseur of corporate bonds. He decides that IndebtedCorp is a pretty good risk, and so he wants to buy up the entire bond issue. He buys everything from Alex and Andrea. From James's perspective, the two bond issues are identical. And yet, if we follow the rule of construction of the one-to-many contract, James may not be able to

115. *See, e.g., id.*

116. *Id.* at 3.

117. *Id.*

118. And indeed, several plaintiffs attempt to argue this. *See infra* notes 119, 122-123 and accompanying text.

identify his rights based on the strict language of the bond issue. Instead, the interpretation of every bond issue would depend upon who supplied the agreement in the original negotiations and whether the parties had specific interactions that would shed additional light on those terms—facts that James might have to expend considerable time and effort to discover.

The situation, of course, magnifies in complexity once you add in what happens in the modern world. Between the issue of the bonds and James's purchase, fifty transactions may take place. Alex sells to Terry and Tony; Tony sells to Betty and Bobby. Bobby then buys half of his bonds from Andrea. At some point, it would be nearly impossible to determine who the original purchaser of a bond was, let alone determine if the identical language resulted in a wildly different contract interpretation.

Because both the bondholder and the debt can be transferred among a large number of parties, uniformity once again matters. Many parties are involved, and while the relationship between initial pairs may be clear, subsequent transfers obscure and destroy any inferences that could have been drawn from the relationships between the parties. The specific ceases to matter; instead, parties determine the worth of the contract by looking at the language on the face of the bonds.

It is ten years and ten thousand transfers later. A purchaser of the Alex/Andrea bonds, when told that the worth of her investments depends not upon the language of the bond contracts but upon the differing relationships between Alex, Andrea, and IndebtedCorp, will respond, "What the devil do Alex and Andrea have to do with me?"

Unlike the Rockbox case, courts have spoken about this particular scenario. Language that has been drafted for a particular bond issue is sometimes construed against the party that supplies the language.[119]

But the presumption against boilerplate language does not hold sway in the bond indenture context. Courts have explained that even though there are several different boilerplate provisions floating around as to the risk that companies can take on,

> there is seldom any difference in the intended meaning [of boilerplate provisions]. The use of standardized language can result in a better and quicker understanding of those provisions and a substantial saving of

119. *See, e.g.*, Fontainebleau Hotel Corp. v. Rosenberg, 121 So. 2d 675, 677 (Fla. Dist. Ct. App. 1960).

> time not only for the draftsmen but also for the parties and all others who must comply with or refer to the indenture . . . .[120]

Because this "time saving" benefits both the draftsmen and bond traders, this principle is distinct from, say, an insurance company's desire to save time in the one-to-many context.[121] The insurance company, of course, wants uniformity because it would ease its calculation of damages and save it time in drafting the document. Here, the benefits of uniformity accrue not just to the person who provides the language, but to everyone interested in trading the bonds.

> A large degree of uniformity in the language of debenture indentures is essential to the effective functioning of the financial markets: uniformity of the indentures that govern competing debenture issues is what makes it possible meaningfully to compare one debenture issue with another . . . without being misled by peculiarities in the underlying instruments.[122]

The uniformity in the language can therefore be extended to a uniformity of interpretation. "Whereas participants in the capital market can adjust their affairs according to a uniform interpretation, . . . the creation of enduring uncertainties as to the meaning of boilerplate provisions would decrease the value of all debenture issues and greatly impair the efficient working of capital markets."[123] In other words, when a large number of transfers potentially interweave party structure into a many-to-many web, uniformity gains in importance. After all, the parties' ultimate intent in constructing the bond agreement is to *maximize* the value of the contract. If that purpose is best served by uniformity of interpretation, uniformity will ultimately control.

It is not only the *number* of transactions that matters in the many-to-many context, but the public nature thereof. Bond debenture agreements often exist for decades. During that time, the debenture issues are relatively liquid and are traded in arms-length transactions by people who know nothing of each other. By the time IndebtedCorp begins to flounder and starts looking for ways to revise its indebtedness, Alex and Andrea are usually well out of the picture. It

120. AM. BAR FOUND., *supra* note 110, at 3.

121. In particular, the time saved is information costs: if all bond issues are interpreted identically, investors will know what they are selling and what they are buying. *Id.* at 1-3.

122. Broad v. Rockwell Int'l Corp., 642 F.2d 929, 943 (5th Cir. 1981).

123. Sharon Steel Corp. v. Chase Manhattan Bank, 691 F.2d 1039, 1048 (2d Cir. 1982).

HeinOnline -- 86 Tul. L. Rev. 553 2011-2012

would be one thing, of course, to insist that Alex should shoulder the burden of ambiguity. It is quite another to insist that the hundred times removed buyer of bond issues should be on the hook. If the goal is to provide incentives for people to avoid ambiguities in drafting, this fails utterly. The incentive in the many-to-many bond debenture context—*do not buy bonds unless you know who supplied the language in the first place*—has less force.

This need for uniformity has thus rendered obsolete the one-to-many contract rules that have been discussed above. "While as a matter of abstract contract law it is proper to construe ambiguities against the drafter of a contract, that tenet of contract law has only limited practical significance in the context of construing an indenture."[124] In addition to the fact that little information is available as to party intent and behavior,

> there is a significant likelihood . . . that different juries, construing identical indentures in an attempt to divine "the intent of the parties," would reach different conclusions. This would indeed be anomalous, since the principal goal of using boilerplate language in such contracts is that there be uniform construction of those provisions.[125]

The bond context thus provides several identifying features of many-to-many contracts. First, in a many-to-many contract, it is believable that a party on any side of the contract could have chosen the initial language, such that a stranger who saw the initial parties shake hands initially would not be able to figure out who had written the terms. Second, in a many-to-many contract, the benefits of the contractual terms adhere in a public or a quasi-public fashion. In the bond context, the benefits of the issue adhere initially to those who back the bonds. But as the bonds are sold and resold the benefits diffuse among members of the public. Finally, the parties to a many-to-many contract pick boilerplate terms not because it is more convenient for one party to do so nor to save the time of negotiations.[126] Instead, the parties choose boilerplate language so that purchasing parties know what they are buying. They recognize that uniform

---

124. *Broad*, 642 F.2d. at 948 n.20.

125. *Id.* In *Broad*, it should be noted that the court found the indentures unambiguous. *Id.* at 955. Despite the high-minded rhetoric quoted above, the *Broad* court dithered around about what it would do if the provisions were really and truly ambiguous. *Id.* at 948 n.20. Other courts hold that boilerplate terms should be interpreted like statutes: their meaning depends only on the contract language and not on the factual relationships between parties. *Sharon Steel*, 691 F.2d at 1048.

126. The parties are actually putting enough money into the bond such that negotiations might be useful.

treatment of bond language is vital. The uniform language is not about saving time on the part of the drafter, but rather about signaling that the parties intend to affect a uniform interpretation.[127]

But is this rule, established in the case of bond indentures, portable to other contexts? Bonds and the GPL share several similarities. Both agreements are designed to effectuate a transfer. In bond issues, both bonds and companies are bought and sold as part of a capital market. In software projects released under the GPL, licenses to intellectual property are passed on to successive holders. In both cases, the number of parties that did not elect contract terms grows rapidly once the boilerplate is chosen. Thus, the initial relationship between contracting parties ceases to be meaningful to latecomers who purchase bonds fifteen years after their inception, or who modify a program that was released under the GPL fifteen years ago.

Likewise, the parties do not choose boilerplate agreements because they do not want to waste time dickering between parties. In that case, one can infer that the party supplying the language intends to accept the burden of whatever may result in construing the language. Instead, the parties choose identical language so that those who wish to accept the terms of the agreement—either by purchasing the bond or by participating in the creation of software under the GPL—know precisely what they are opting to get. In other words, the parties intend uniformity.

This Article thus suggests that the GPL should be treated much like a bond indenture agreement: it is an agreement that should be evaluated uniformly, such that the general language of the accompanying license should trump the specific. That is, after all, what the parties intended.

V. CONCLUSION

When uniform language no longer signals that one party does not want to negotiate but instead signals that both parties want the contract to be interpreted uniformly, specific rules that allow for nonuniformity fail to usefully capture party intent. When the benefits of the contract cease to adhere to the person who provides the language, the rule that ambiguities should be construed against the party that supplied the

127. This can even be true in the insurance context. *See* Hazel Glenn Beh, *Reassessing the Sophisticated Insured Exception*, 39 TORT TRIAL & INS. PRAC. L.J. 85, 104-05 (2003) (arguing that standardized language among sophisticated insureds is intended to be uniformly interpreted).

HeinOnline -- 86 Tul. L. Rev. 555 2011-2012

language provides no motive for the person who provides the language to strive for clarity.

Many-to-many contracts have cropped up in at least two areas in our economy. The first, as exemplified by the bond instance, is in financial markets. As transactions grow, and as certain kinds of transactions become more and more available to the public, private agreements governing property that can easily be traded take on semipublic meaning.

The second area is less financial in nature. People are becoming increasingly aware that government provides default rules and regimes. Coupled with that awareness is the notion that those who dislike the defaults can contract around them. Seen in this light, the GPL contracts around intellectual property rules that some find disturbing.

The GPL is not unique in this regard. For instance, a certification mark that has been recently introduced attempts to contract around default employment rules. Exasperated with the rate of legal protection provided for gay and lesbian workers, Professors Ian Ayres and Jennifer Brown have suggested that a "fair employment" mark be licensed to employers who promise not to discriminate against gay and lesbian employees—and who back that promise with a guarantee that employees who have been discriminated against can sue for damages.[128] Thus, the benefits of the contract between the licensor and the company flow not to the licensor directly, but to consumers—who get more information—and employees—who benefit in guarantees of nonharassment. The end result is that the private agreements between the mark's licensor and the licensees may have public effect similar to that discussed here. Ayres and Brown assume that their license would operate like an opt-in statute, with uniform effect and interpretation.[129] But if it were interpreted as a one-to-many contract, that might not be the case.

Likewise, imagine a music file released under a Creative Commons license. It might be resampled by another user; the resampling might then be used in a video, which might be excerpted in another video or parodied by another user.[130]

---

128. Ian Ayres & Jennifer Gerarda Brown, *Mark(et)ing Nondiscrimination: Privatizing ENDA with a Certification Mark*, 104 MICH. L. REV. 1639 (2006).

129. *Id.* at 1648-50.

130. See, for instance, music and sound files shared and sampled, or news blogs that release under another many-to-many contract, the Creative Commons share-and-share alike license.

But many-to-many contracts can also be commercially oriented. For instance, MPEG-2 is a standard for encoding digital video—in fact, it is the standard that underlies the movies that are purchased by end users on DVD. It was a standard that was created by a consortium of patent holders. The patent pool is a set of license agreements which requires users to grant back improvements to the technology to the pool as a whole.[131]

As our markets become increasingly interconnected, and as people become increasingly creative in formulating contractual alternatives to statutory defaults, the number of many-to-many contracts will only increase. Contract interpretation should be ready to interpret those contracts in a manner consistent with their structure. When private agreements give rise to many-to-many effects, a consistent public interpretation should apply.

131. Wan-Hsin Liu & Tillmann Schwörer, *Open Innovation and Access to Knowledge*, GLOBAL ECON. SYMP. 2011, at 11 (Apr. 15, 2011), http://www.global-economic-symposium.org/solutions/the-global-economy/open-innovation-and-access-to-knowledge/background-paper-open-innovation.

HeinOnline -- 86 Tul. L. Rev. 557 2011-2012

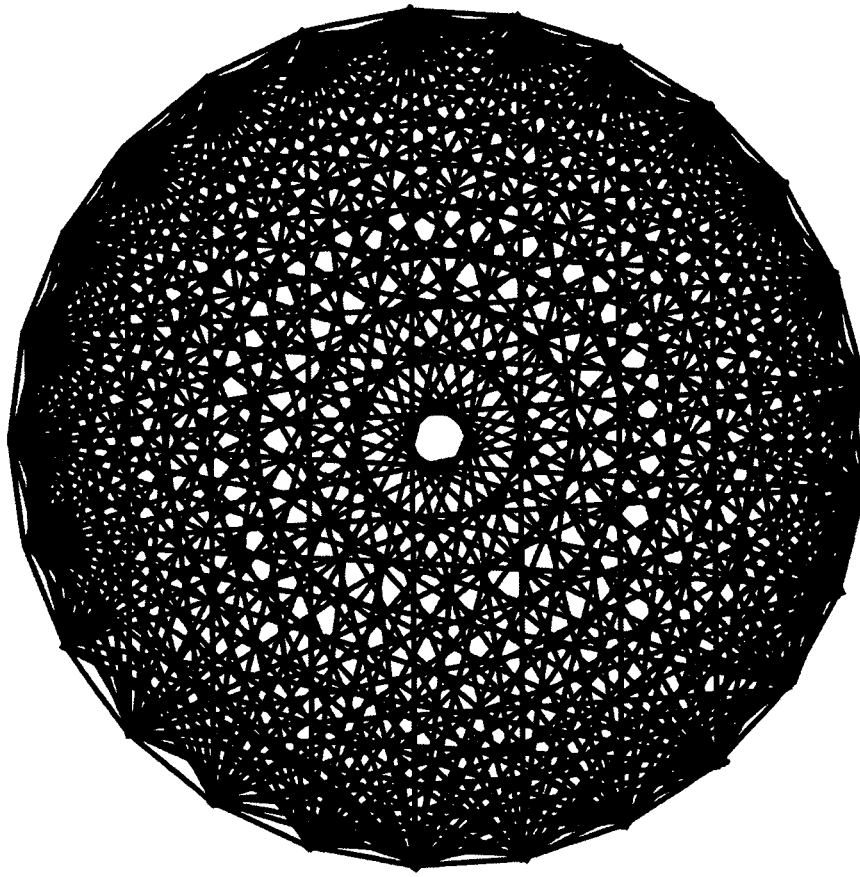

Figure 1: Depicted is the web of contract interactions among the Rockbox developers. Each vertex indicates a developer; every line connecting vertices indicates that developer has licensed code from the other. In most cases, because developers contribute to the project piece by piece and day by day, and download new versions of the code in order to continue development, the licenses flow in both directions. Freese must license code from Stenberg, but Stenberg will need a license back from Freese.